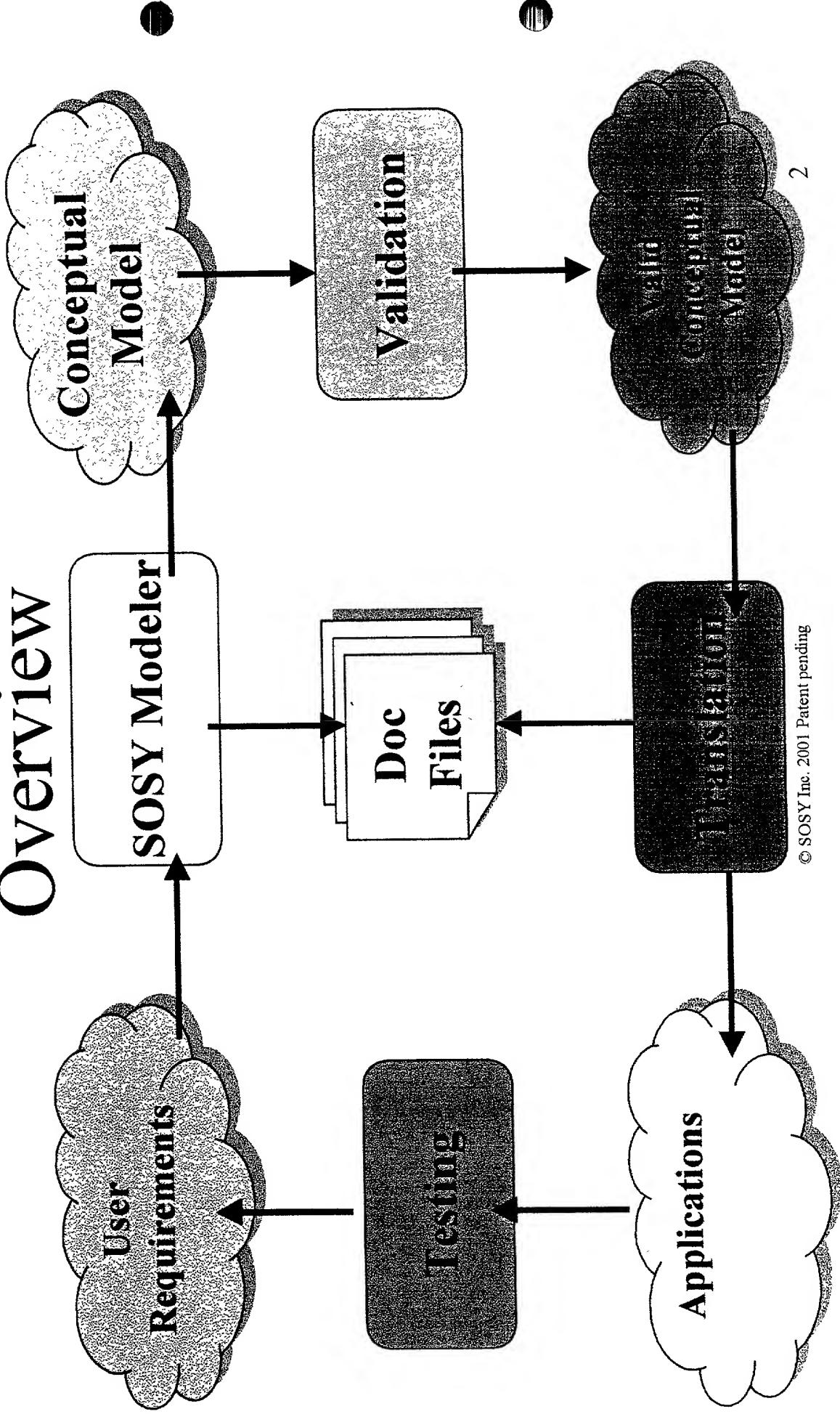


Summary

- **MANUFACTURING**
- **PACKAGING**
- **DISINFECTION**
- **DISINFECTION LOGIC**

Overview



Conceptual Modeling Phase

CARE Technologies, S.A.

Index

- Intro
- Overview
- Phase 0. Requirements elicitation.
- Phase 1. Classes identification.
- Phase 2. Relationships between classes.
- Phase 3. Filling classes' details.

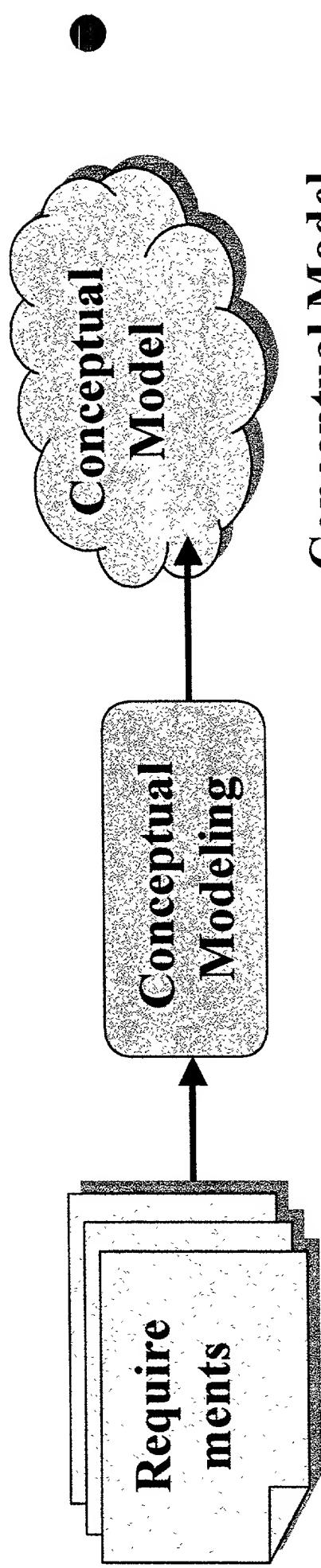
Index

- Phase 4. Express evaluations.
- Phase 5. Agent relationships.
- Phase 6. State Transition Diagram.
- Phase 7. Presentation Model.

Intro

- Conceptual Modeling Phase is a process of systematically & precisely description of the system to build, using:
 - Graphical UML compliant diagrams.
 - Constraints and semantics in a formal non-ambiguous language.
 - This phase is assisted by an integrated Modeler tool.

Overview



Requirements

- Specifications
- Documents
- Interviews
- Reports
- Other info. sources

Conceptual Model

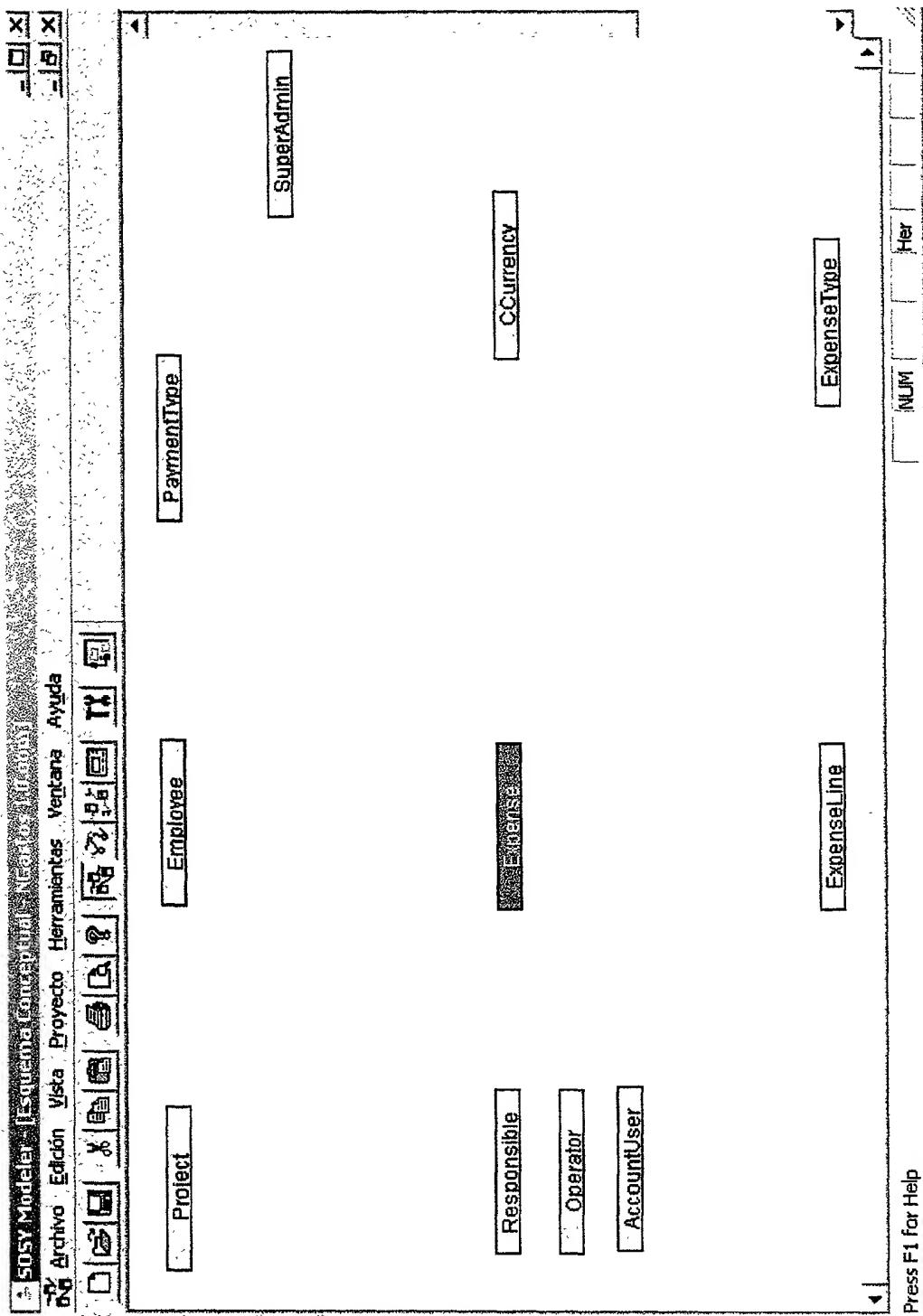
- Classes
- Relationships
- Attributes
- Services
- ...

Expressed in a non-ambiguous language.

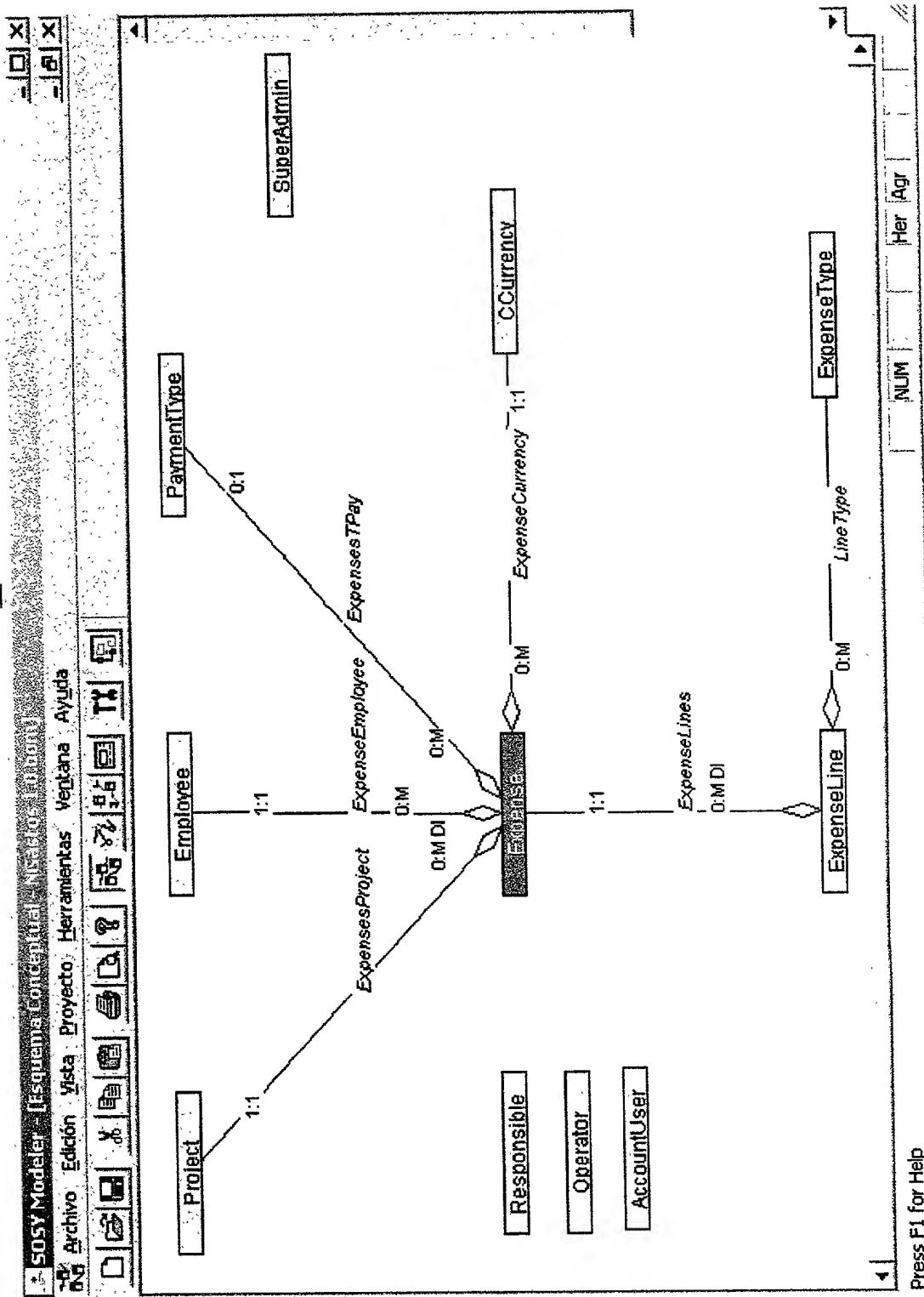
Phase 0. Requirement elicitation.

- **Gathering the system requirements.**
 - By meetings & interviews with customers, experts and final users.
 - By collecting reports, or documents expressing the system how-to and using tools.
 - Obtaining a coherent set of information as input to the next phase.

Phase 1. Classes identification.



Phase 2. Relationships between classes.



Phase 3. Filling classes' details.

Phase 3. Filling classes' details.

Clase

Atributos | Servicios | Restricciones | Agentes | Transacciones | Relaciones | Generales | Generales

Balance

Fórmulas de Derivación

Condición

Fórmula

Observaciones

Añadir

Modificar

Borrar

Aceptar

Cancelar

Clase: Expense

Phase 3. Filling classes' details.

Clase:

atributos | servicios | restricciones | agentes | transacciones | relaciones | generalidades |

Eventos y transacciones

Servicios:

Nombre	Características	Tipo dato
newexpense	<input type="button" value="Añadir"/>	Expense
deleteexpense	<input type="button" value="Añadir"/>	String
destroy	<input type="button" value="Añadir"/>	Real
close	<input type="button" value="Añadir"/>	Real
authorize	<input type="button" value="Añadir"/>	Real
approve	<input type="button" value="Añadir"/>	Real
pay	<input type="button" value="Añadir"/>	Real
rejectautho	<input type="button" value="Añadir"/>	Real
rejectpayment	<input type="button" value="Añadir"/>	Real
InstPaymentType	<input type="button" value="Añadir"/>	Real
BankPaymentType	<input type="button" value="Añadir"/>	Real
DELETEALL	<input type="button" value="Añadir"/>	Real
TPAY	<input type="button" value="Añadir"/>	Real

Parámetros:

Nombre	Tipo dato
o_ThisExpense	Expense
o_Cause	String
o_Advances	Real
o_Exchange	Real

Attributes

Nombre	Tipo dato
Nombre:	<input type="text"/>
Imágen:	<input type="text"/>
Uso interno:	<input type="checkbox"/>
Alias:	<input type="text"/>
Mensaje de Ayuda:	<input type="text"/>
Observaciones:	<input type="text"/>

Observaciones:

Valor por defecto:

Observaciones:

Phase 3. Filling classes' details.

Acción

Clase/Id:

Expense

Agentes:

Servicio:

Párametros:

Inicializar:

Observaciones:

Create Paquete en la Transacción

Aceptar Cancelar

Clase: Expense

Phase 3. Filling classes' details.

Clase: Expense

Atributos | Servicios | Restricciones | Agentes | Transacciones | Relaciones | Generaldades | Heredadas

Clase: Expense

Estáticas

Exchange > 0

Fórmula: Exchange > 0

Mensaje De Error: Exchange must be greater than zero

Dinámicas

Fórmula: Oper. Temporal

Mensaje De Error: Oper. Temporal

Fórmulas

Fórmula: Oper. Temporal

Mensaje De Error: Oper. Temporal

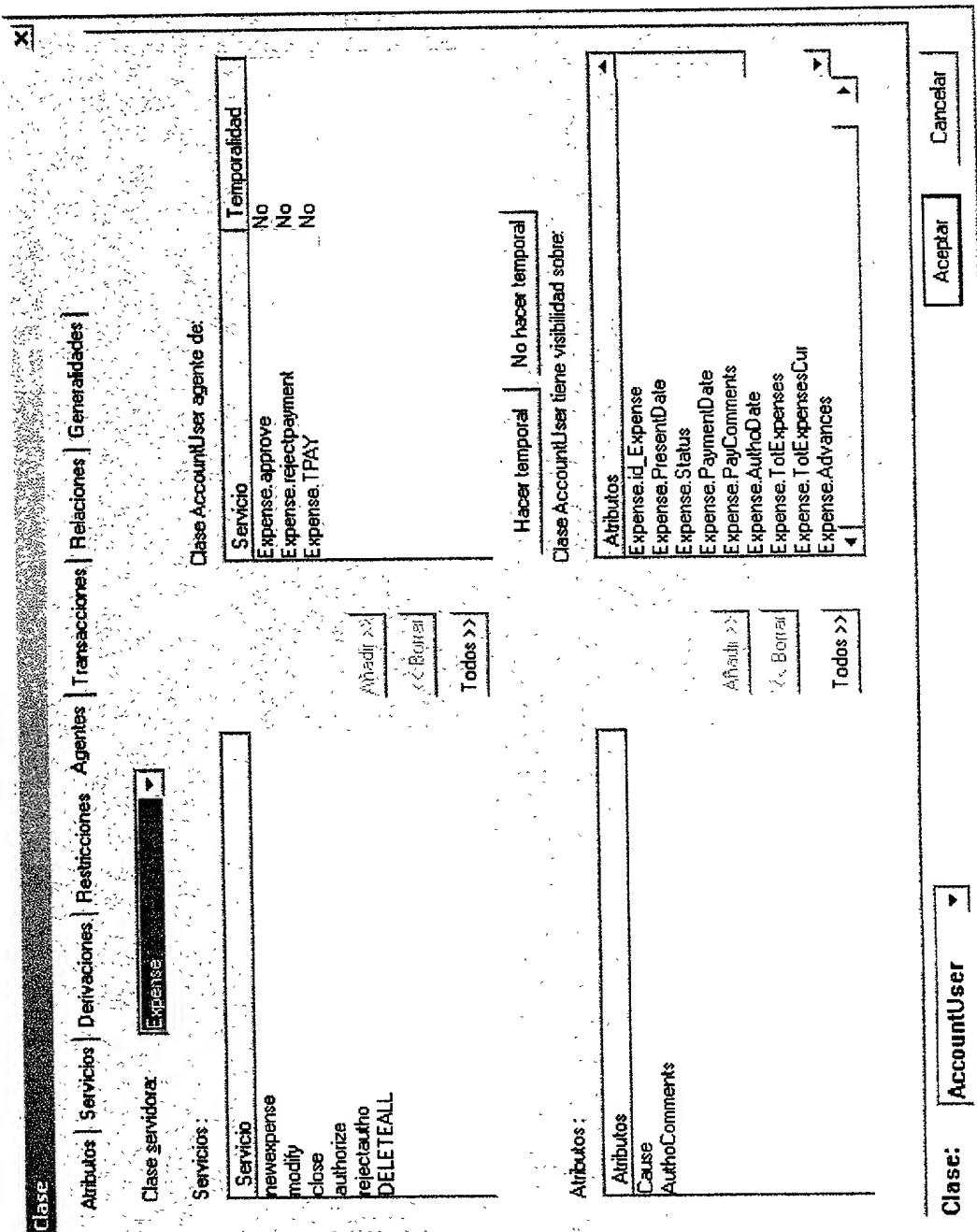
Añadir | Modificar | Borrar | Heredadas

Aceptar | Cancelar

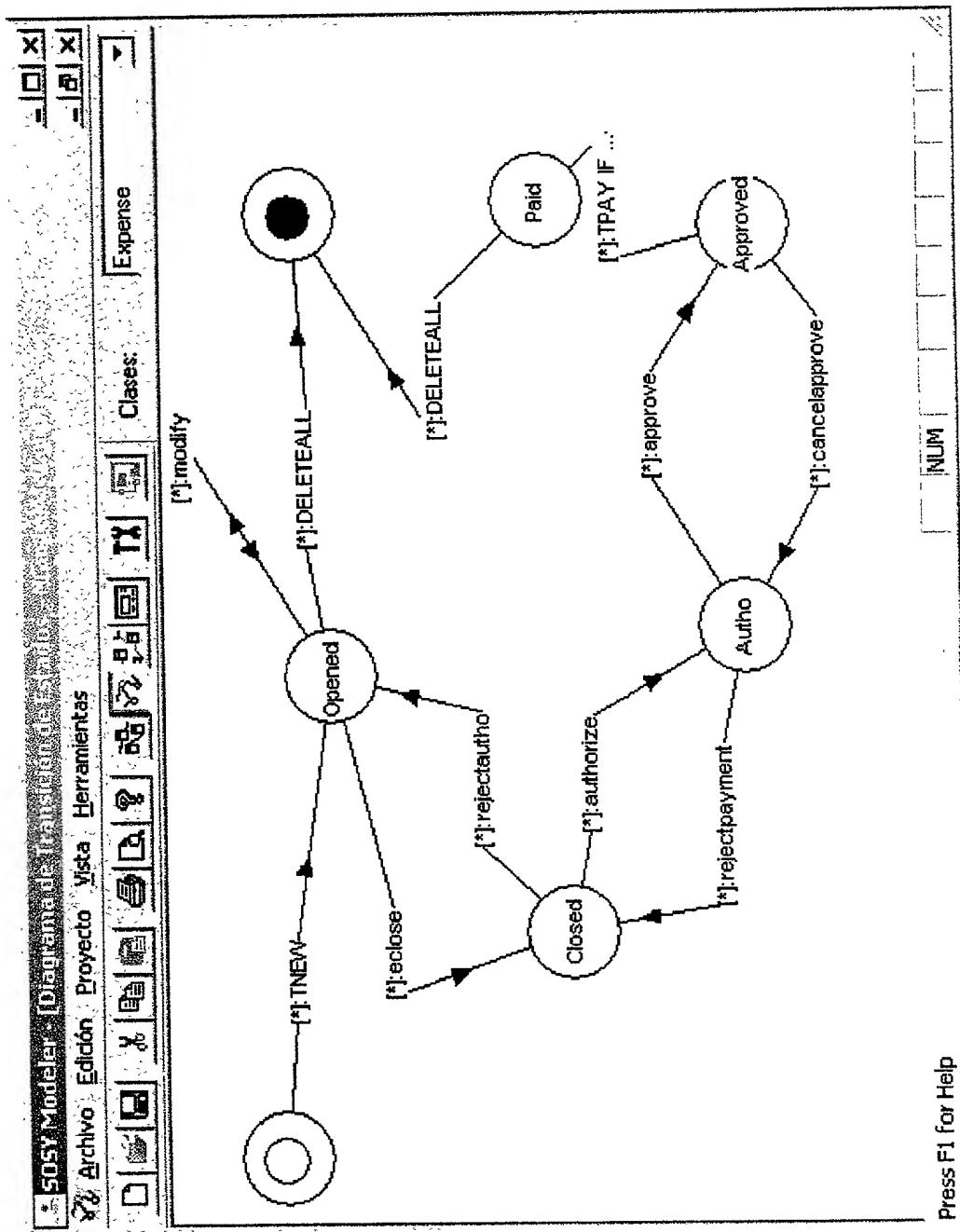
Phase 4. Express evaluations.

Modelo Funcional							
Clase: <input type="text" value="Expense"/>	Atributo: <input type="text" value="Cause"/>						
<input type="button" value="Aceptar"/> <input type="button" value="Cancelar"/>							
<input type="button" value="Añadir"/> <input type="button" value="Modificar"/> <input type="button" value="Borrar"/>							
<table border="1"> <thead> <tr> <th>Evento</th> <th>Efecto</th> <th>Condición</th> </tr> </thead> <tbody> <tr> <td>modify</td> <td><input type="text" value="P_Cause"/></td> <td></td> </tr> </tbody> </table>		Evento	Efecto	Condición	modify	<input type="text" value="P_Cause"/>	
Evento	Efecto	Condición					
modify	<input type="text" value="P_Cause"/>						
<input type="checkbox"/> Cardinal <input checked="" type="checkbox"/> De Estado <input type="checkbox"/> De Situación							
Detalles de Evaluación							
Evento:	<input type="text" value="modify"/>						
<input type="checkbox"/> Inferir para el resto de atributos							
Condición de evaluación:							
IF:	<input type="text" value="IF: Efecto del evento: P_Cause"/>						

Phase 5. Agent relationships.



Phase 6. State Transition Diagram.



Phase 6. STD Preconditions

Transición	
Origen:	<input type="text" value="Approved"/>
Destino:	<input type="text" value="Paid"/>
Detalles	<input type="checkbox"/>
Agentes:	<input type="checkbox"/> AccountUser <input type="checkbox"/> SuperAdmin
Precondición:	
<input type="text" value="TPAY"/>	
<input type="text" value="Servicio:"/>	
<input type="text" value="Balance > 0 OR ps_ReturnAdvance = TRUE"/>	
Condición de control:	
<input type="checkbox"/>	
Mensaje en caso de Error:	
<input type="text" value="Check the advanced money excess"/>	

Phase 7. Presentation Model.

Conjunto de Visualización

Nombre: ICv_Expense

Atributos a visualizar:

Atributo	Tipo dato	Añadir	Eliminar >>
Project.ProjectName	String	<input type="button" value="Subir"/>	<input type="button" value="Bajar"/>
Employee.EmpName	String	<input type="button" value="Subir"/>	<input type="button" value="Bajar"/>
Employee.EmpSur...	String	<input type="button" value="Subir"/>	<input type="button" value="Bajar"/>
Status	Int	<input type="button" value="Subir"/>	<input type="button" value="Bajar"/>
AuthoDate	Date	<input type="button" value="Subir"/>	<input type="button" value="Bajar"/>
PaymentDate	Date	<input type="button" value="Subir"/>	<input type="button" value="Bajar"/>
TotExpenses	Real	<input type="button" value="Subir"/>	<input type="button" value="Bajar"/>
Balance	Real	<input type="button" value="Añadir"/>	<input type="button" value="Agrega"/>

Atributos:

Atributo	Tipo dato	Atributo	Tipo dato
Cause	String	AuthoComments	String
AuthoDate	Date	PaymentDate	Date
AuthoComments	String	PayComments	String
PaymentDate	Date	TotExpenses	Real
PayComments	String	TotExpensesCur	Real
TotExpenses	Real	Advances	Real
TotExpensesCur	Real	AdvancesCur	Real
Advances	Real	Exchange	Real
AdvancesCur	Real	Balance	Real
Exchange	Real	BalanceCur	Real

Clase: Expense

Phase 7. Presentation Model.

Filtro					
Filtro:	fit_Expense	Alias:	Expense Reports	Borrar	Limpiar
Fórmula:	<input type="text" value="Project = vf_Project AND Employee = vf_Employee AND PresentDate >= vf_DateInIssue AND PresentDate <= vf_DateEndIssue AND AuthoDate >= vf_DateInApp AND AuthoDate <= vf_DateEndApp AND PaymentDate >= vf_DateInPay AND PaymentDate <= vf_DateEndPay AND"/>				
Observ:	<input type="text" value=""/>				
Variables					
Nombre	Alias	Tipo dato	Tipo estilo	Estilo	Nueva
vf_Project	Project	Project	Sel. Población		
vf_Employee	Employee	Employee	Sel. Población		
vf_DateInIssue	InitialIssuing Date	Date			
vf_DateEndIssue	FinalIssuing Date	Date			
vf_DateInApp	Initial Approving D...	Date			
Tipo					
<input type="radio"/> Simple <input type="radio"/> Objeto-valorizado		Nombre:	<input type="text"/>	Estilo de introd.:	<input type="text"/>
		Alias:	<input type="text"/>	Estilo de selección:	<input type="text"/>
		Tipo de dato:	<input type="text"/>		
Clase: Expense					
<input type="button" value="Aceptar"/> <input type="button" value="Cancelar"/>					

Conceptual Model Validation

CARE Technologies, S.A.

Index

- Intro
- Overview
- Validation Degrees
 - Partial Validation
 - Total Validation

Index

- Validation Types
 - Elements of the Conceptual Model
 - Formulas of the Conceptual Model (Syntax)
- Validation Trees
 - Nodes
 - Leaves
- Example

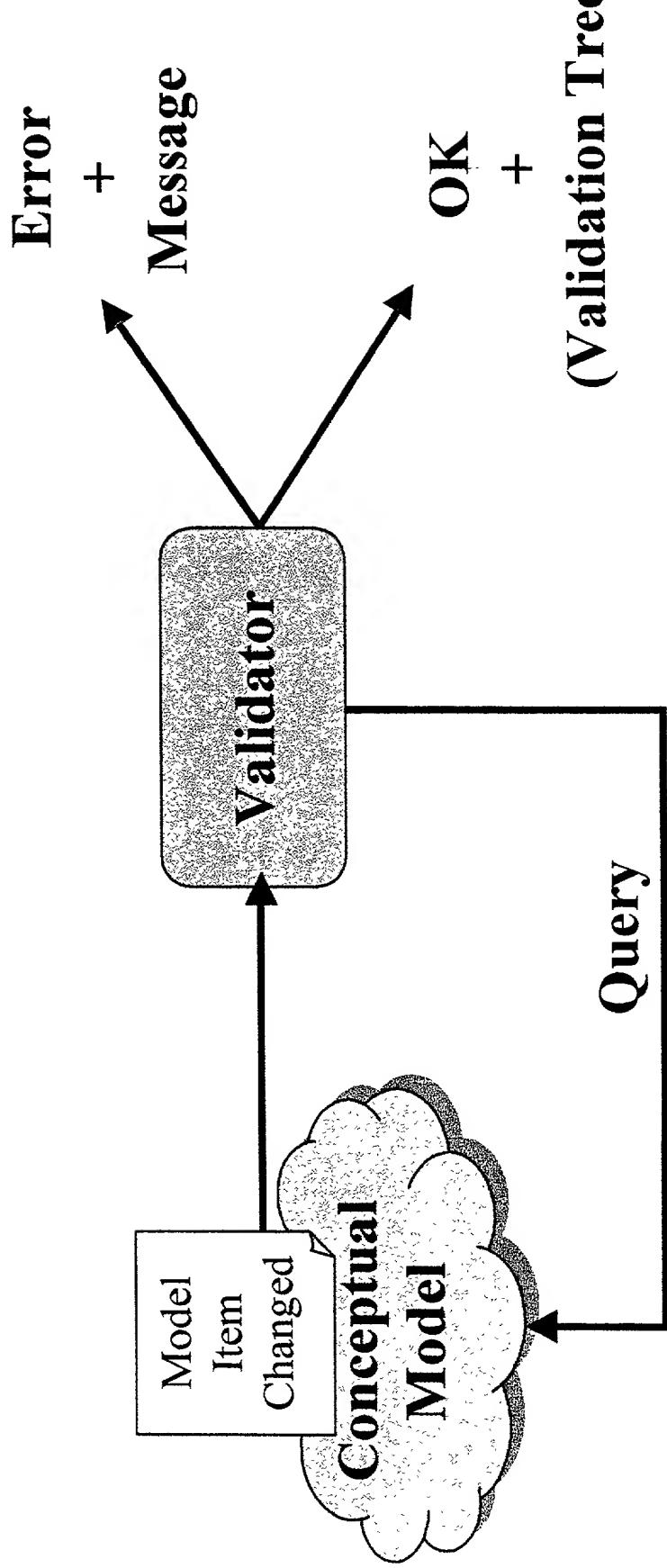
Intro

- Conceptual Model Validation is the process by which a conceptual model or a modification of it is proven to be valid:
 - Correct
 - Non Ambiguous
 - Non Contradictory
 - Complete
 - Every concept is fully specified
- Validation process checks the representation of requirements in Formal Specification Language to be valid

Validation Degrees

- Partial Validation
 - That of a single element of the Conceptual Model.
 - Happens whenever an element is added, modified or deleted.

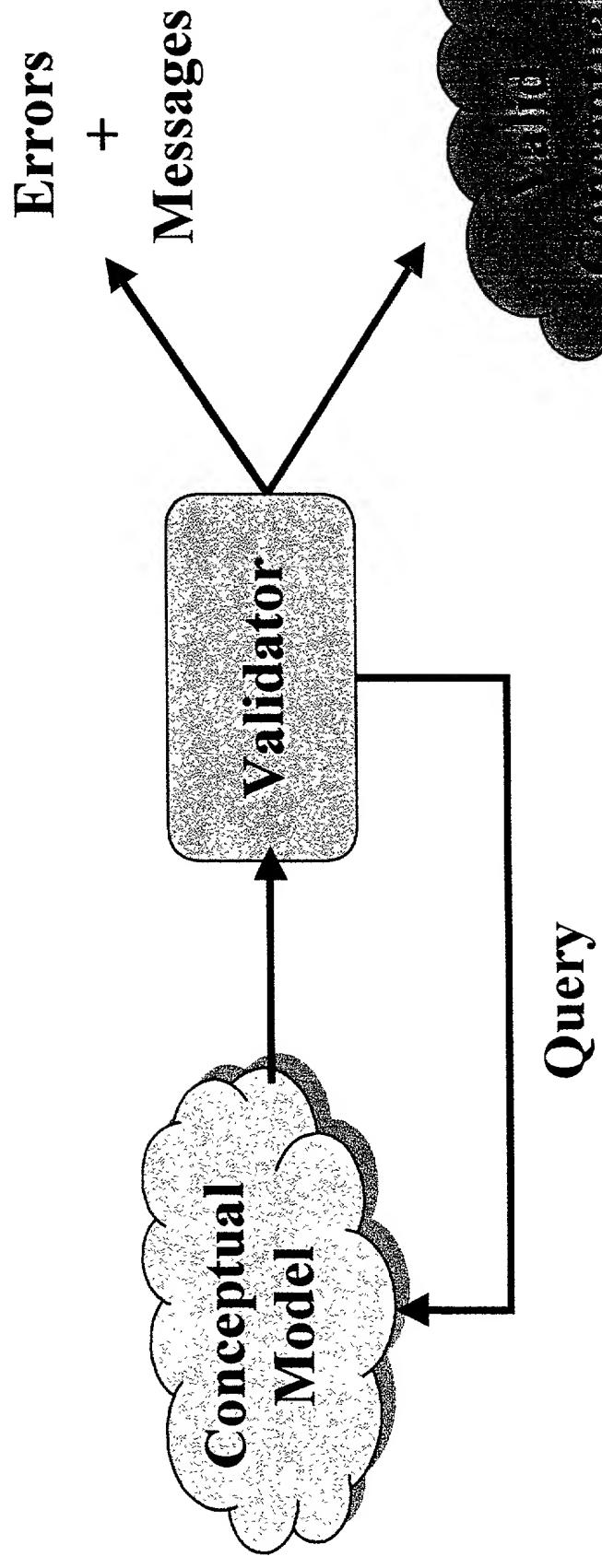
Partial Validation Overview



Validation Degrees

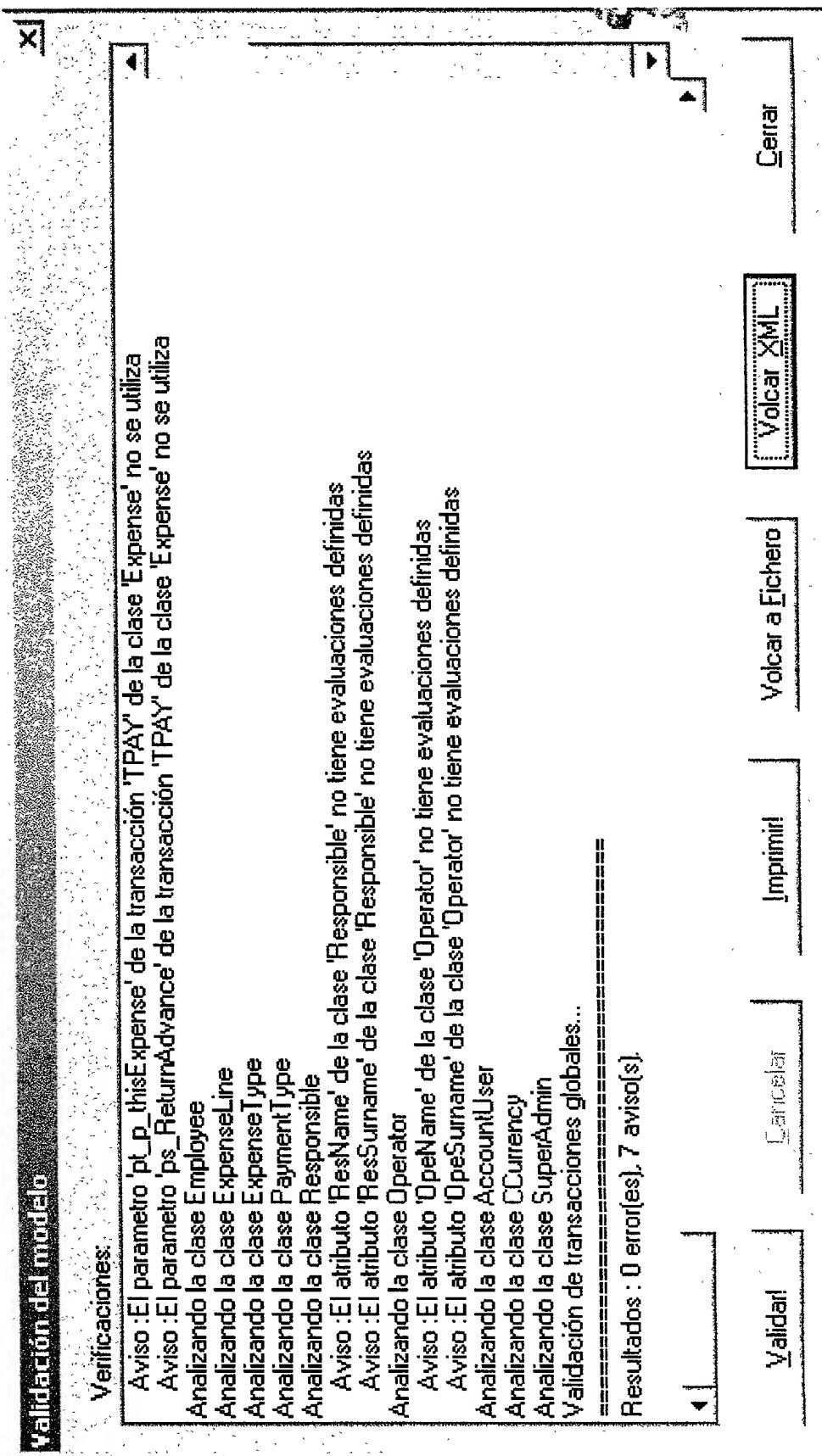
- Total Validation
 - That of the whole Conceptual Model.
 - Happens by request.
 - Must happen prior to any translation process.
 - Takes advantage of partial validations already performed.

Total Validation Overview



© SOSY Inc. 2001 Patent pending

Total Validation Example



Validation Types

- Elements of the Conceptual Model
 - Ensure the properties of an element (except formulas) are correct and complete.
 - Conditions that must hold depend on the type of element and the property being validated.
 - Examples:
 - Class Name is unique in a Conceptual Model.
 - Attribute Name is unique in its Class (but not in a Conceptual Model)

Validation Types

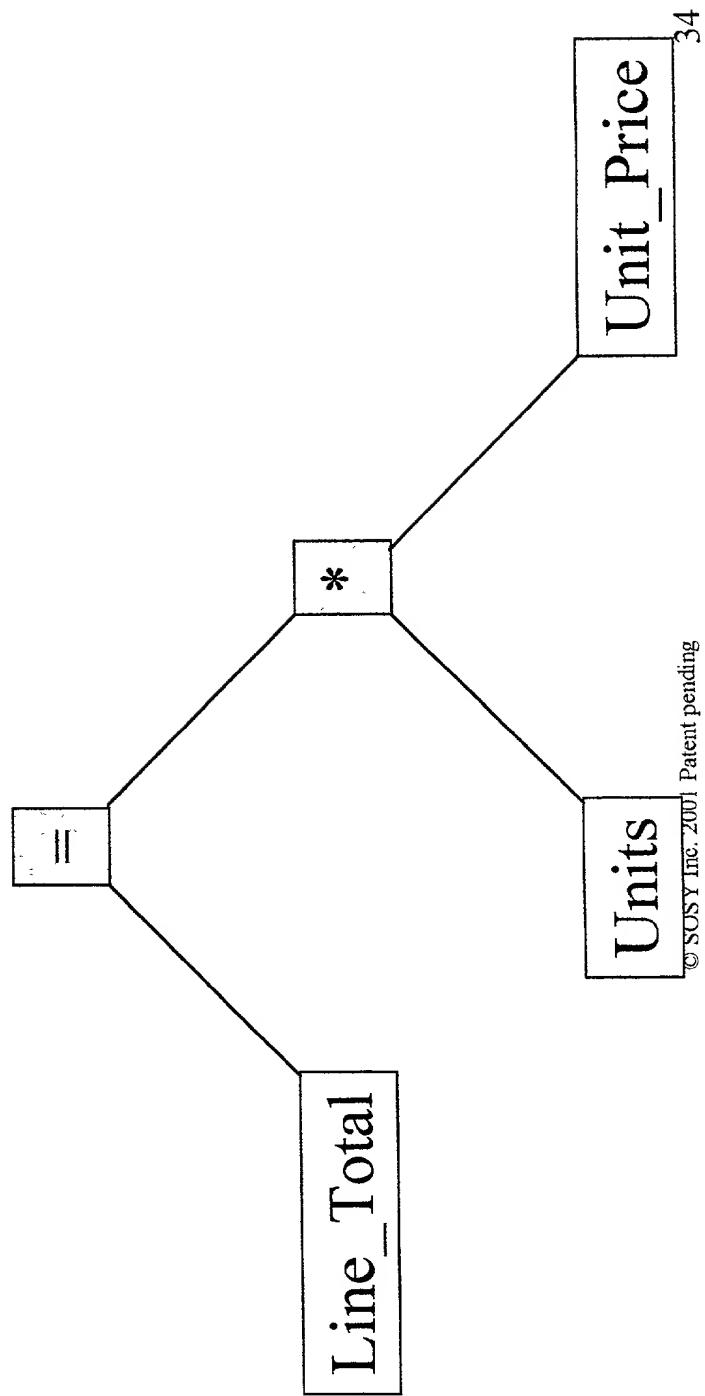
- **Formulas of the Conceptual Model**
 - Ensure the formulas of the Conceptual Model are correct and complete.
 - Syntactical and Semantical Validation according to an extended Formal Specification Language grammar.
 - Input:
 - Formula expression
 - Formula Type (precondition, valuation, ...etc.)
 - Formula Context (class name, service name, ...etc.)
 - Output:
 - Error Message (validation did not pass)
 - Validation Tree (validation the 2007 part pending passed)

Validation Trees

- Binary Tree representation of a correct formula.
- Tree consists of Nodes and Leaves.
- Nodes
 - Represent operators
 - Can have one or two “branches” (binary)
 - Branches can again be nodes or leaves
- Leaves
 - Represent operands
 - Have no branches

Example

- $\text{Line_Total} = \text{Units} * \text{Unit_Price}$



© SOSY Inc. 2001 Patent pending

34

Documentation Translation

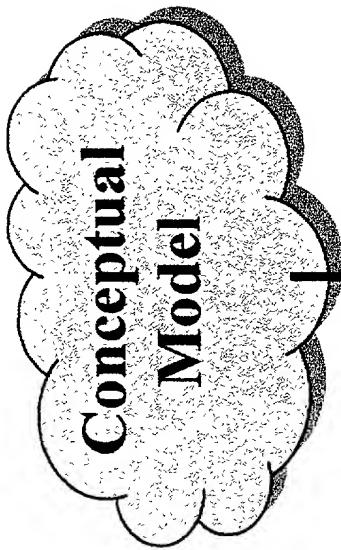
CARE Technologies, S.A.

Index

- Intro
- Overview
- Output Detail
 - Document Types
 - Document Formats
- Translation
 - CM Subset of Interest
 - Translation Process
 - Remarks
- Example

Intro

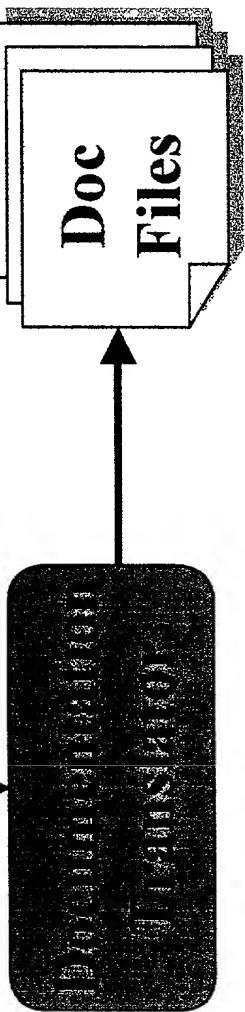
- Documentation Translation is the process to obtain, from a Conceptual Model, documentation on the system it represents.
- Documentation can have several degrees of detail and be focused on different aspects, thus obtaining different documentation formats from the same Conceptual Model.



Overview

Document Type

- Help
- Full
- General
- User Help Manual
- Project Report
- Test Report
- Multifile HTML
- Single File HTML
- ASCII Text
- LaTeX
- RTF
- ^{© SOSY Inc. 2004 Peter Pernig} Compiled HTML



Output Detail

- Document Types
 - Help
 - Description of each Class, its Attributes, Services and Population Selection Filters.
 - Full
 - Full description of a Conceptual Model
 - Aimed at analysts.
 - General
 - Description of each Class Attributes, Identification Function, Services, Aggregation Relationships and Specialization Relationships.

Output Detail

- Document Types
 - User Help Manual
 - Both Help Manual and Contextual Help (F1 key).
 - Intended for Operation Manual.
 - Integration with User Interface applications.
 - Project Report
 - Description of each Class Attributes and Services.
 - Test Report
 - Description of each Class Services.
 - Intended for Testing purposes.

Output Detail

- Document Formats
 - Multifile HTML
 - One HTML page per concept.
 - Recommended for navigable help.
 - Single File HTML
 - One single HTML page.
 - Recommended for printing.
 - ASCII Text
 - Single, plain ASCII text file.

Output Detail

- Document Formats
 - LaTeX
 - Single, LaTeX text file.
 - RTF
 - Single, RTF text file.
 - Compiled HTML
 - Same as Multifile HTML plus header files to be used by HTML Help Workshop compiler.
 - Recommended for contextual help.
 - Searching and Indexing facilities usage from browsers.

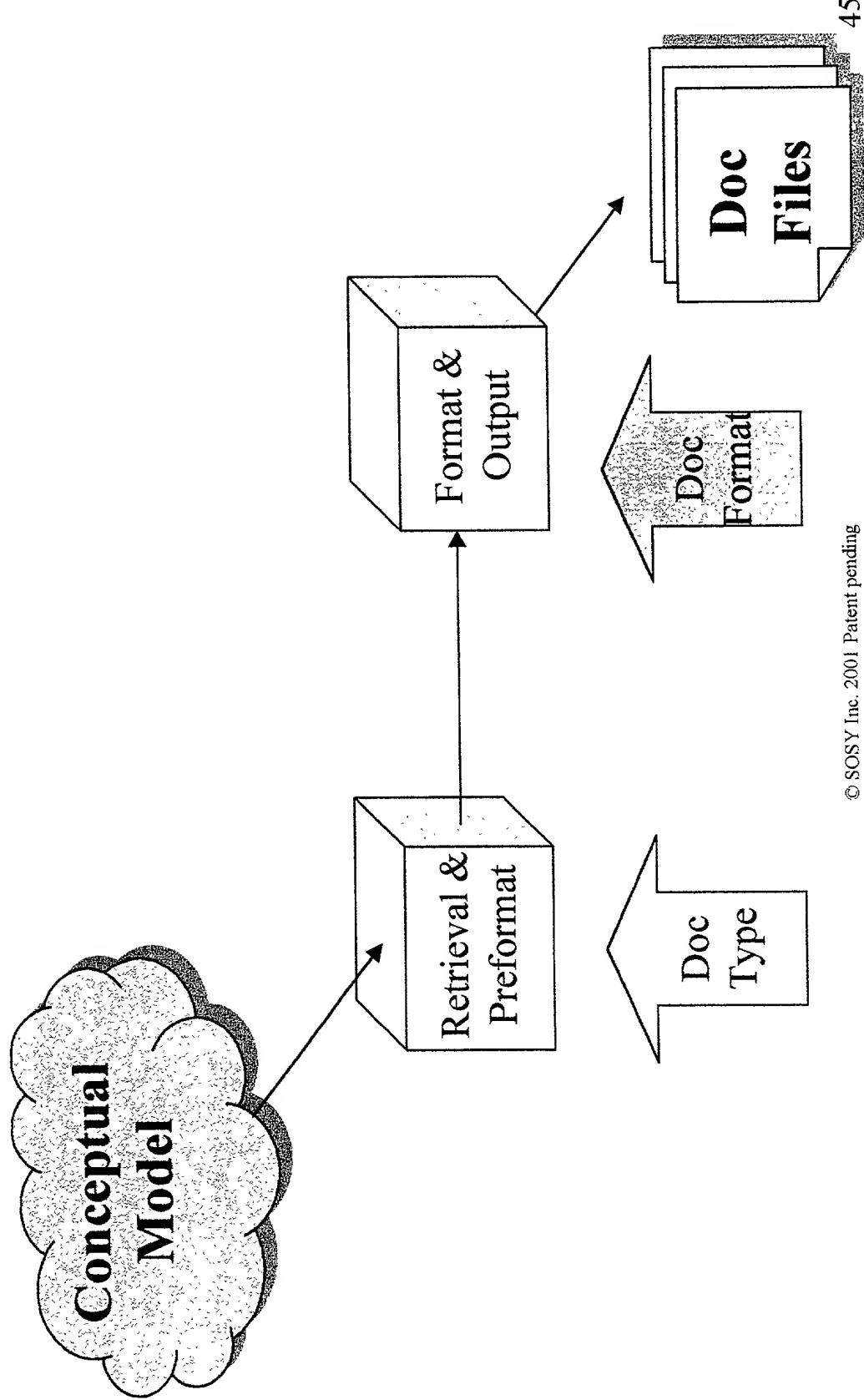
Translation

- Conceptual Model Subset of Interest
 - Subset of Interest depends on Document Type.
 - Usual elements:
 - Classes
 - Attributes
 - Relationships
 - Services & Arguments
 - Intensive use of analysis information.

Translation

- Translation Process
 - Read information from Conceptual Model and format it for output.
 - Two phases:
 - Information retrieval and pre-formatting.
 - Depends on Document Type
 - Independent from Document Format
 - Information output.
 - Depends on Document Format.
 - Independent from Document Type.

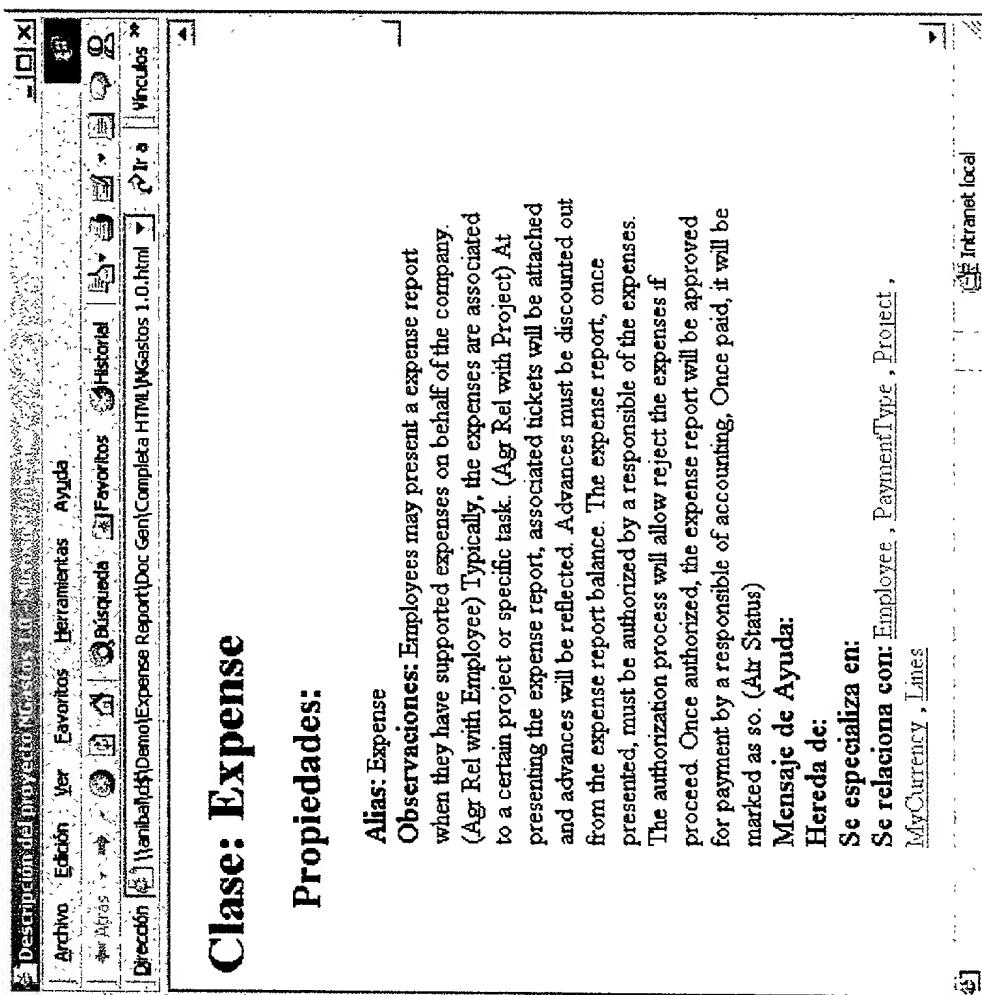
Translation Phases



Translation

- Remarks
 - Conceptual Model needs not to be valid (in terms of completeness and correctness) but it is always non-ambiguous.
 - The richer the analysis information, the richer the documentation.
 - Easily extensible
 - New Document Types
 - New Document Formats

Example



Persistence Relational Database Translation

CARE Technologies, S.A.

Index

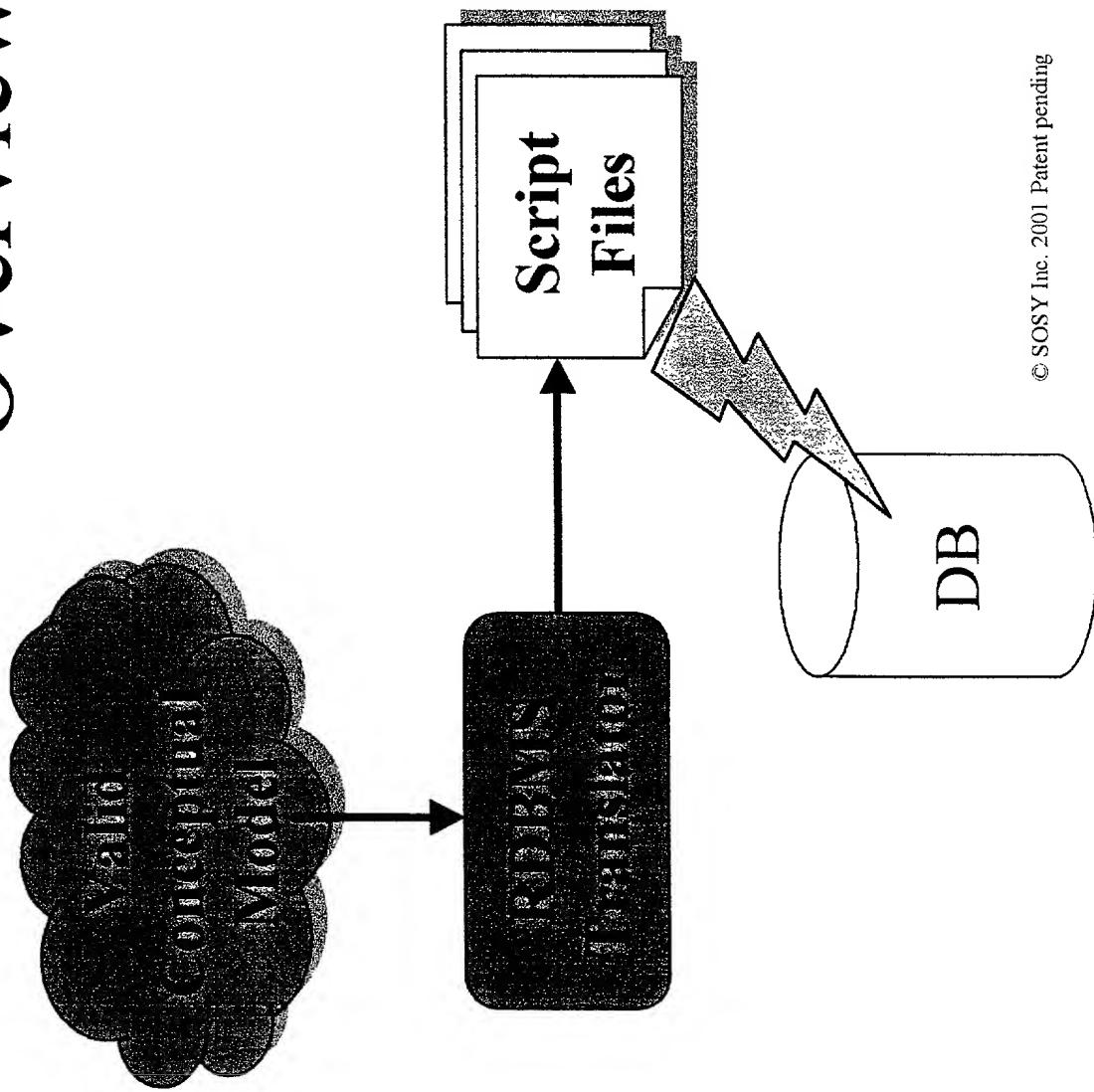
- Intro
- Overview
- Output Detail
- Translation
 - CM Subset of Interest
 - Translation Processes
- Example

Intro

- Persistence Relational Database Translation
is the process of creating a Relational Database from a certain subset of information in the Object Model of a valid Conceptual Model.
- Output script files are used to create a relational database using structured query language (SQL).

Overview

- Creates
- Primary Keys
- Foreign Keys
- Indexes
- Drop Creates
- Drop Primary Keys
- Drop Foreign Keys
- Drop Indexes



Output Detail

- Creates
 - Creation of Tables and Fields
- Primary Keys
 - Creation of Primary Keys as constraints on each table
- Foreign Keys
 - Creation of Foreign Keys as constraints on each table
- Indexes
 - Creation of Indexed on each table

Output Detail

- Drop Creates
 - Deletion of Tables
- Drop Primary Keys
 - Deletion of Primary Key Constraints
- Drop Foreign Keys
 - Deletion of Foreign Key Constraints
- Drop Indexes
 - Deletion of Indexes

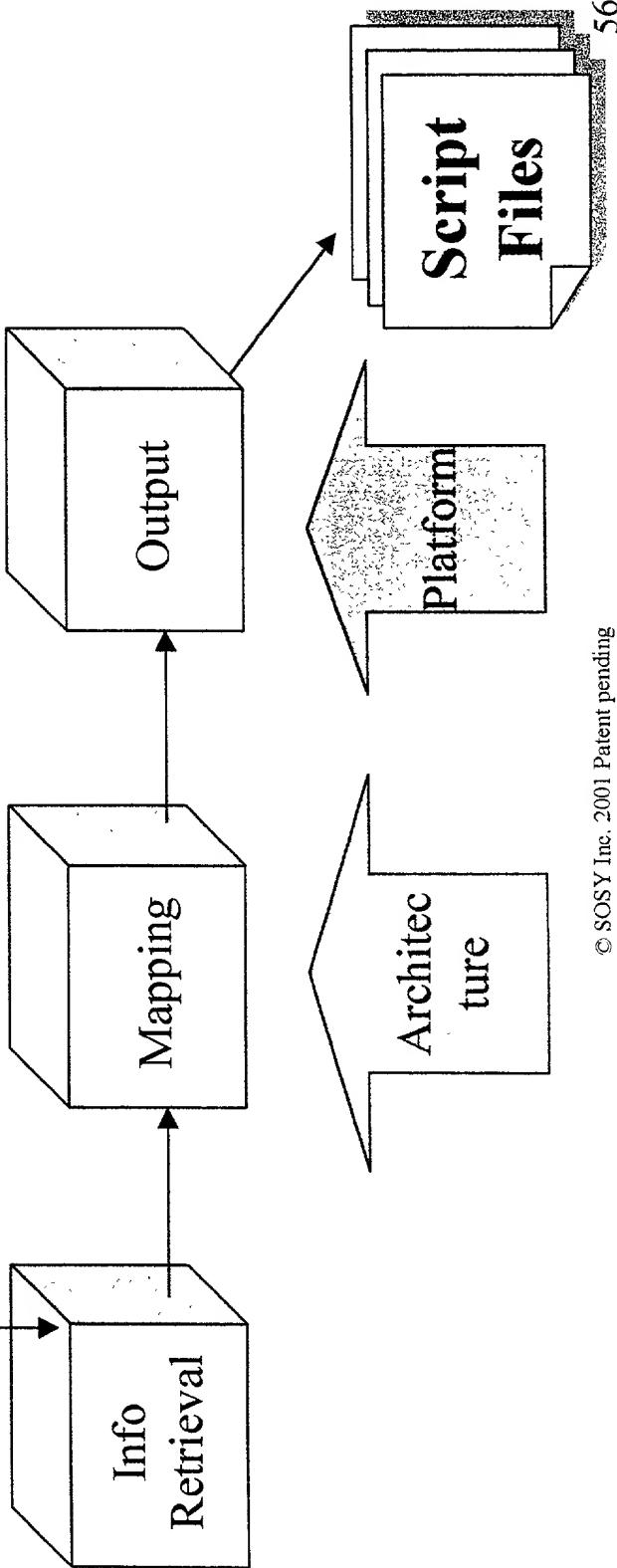
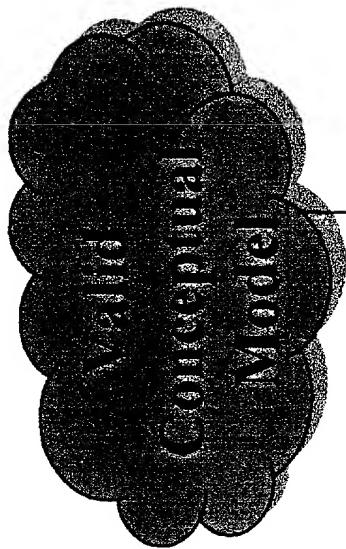
Translation

- Conceptual Model Subset of Interest
 - Object Model
 - Classes
 - Attributes
 - Identification Functions
 - Aggregation Relationships
 - Inheritance Relationships

Translation

- **Three phases:**
 - Information retrieval.
 - Independent from persistence architecture.
 - Fixed architecture mapping.
 - Depends on persistence architecture.
 - Information output.
 - Targeted for Standard ANSI SQL 92 RDBMS.
 - Script files depends on the platform's SQL syntax of RDBMS manufacturer.
 - May depend on platform specifications to make use of manufacturer extensions and tuning.

Translation Phases



Translation

- Translation Processes. Mapping:

- Class → Table
- Non-derived Attribute → Field
- Identification Function → Primary Key
- Univaluated Relationship → Foreign Key
- Univaluated Relationship → Index
- Multivaluated Relationship → Table
- Inheritance Relationship → Foreign Key

Example

Create table script in SQL for Expense class

```
CREATE TABLE Expense (
    fk_Project_1 int NOT NULL ,
    id_Expense int NOT NULL ,
    fk_Employee_1 CHAR(10) NOT NULL ,
    fk_MyCurrency_1 CHAR(5) NOT NULL ,
    fk_PaymentType_1 CHAR(5) NULL ,
    PresentDate datetime NOT NULL ,
    Status int NOT NULL ,
    Cause VARCHAR(255) NOT NULL ,
    AuthorDate datetime NULL ,
    AuthoComments VARCHAR(255) NULL ,
    PaymentDate datetime NULL ,
    PayComments VARCHAR(255) NULL ,
    Advances DECIMAL(10, 6) NOT NULL ,
    Exchange DECIMAL(10, 6) NOT NULL );
```

Business Logic Translation

CARE Technologies, S.A.

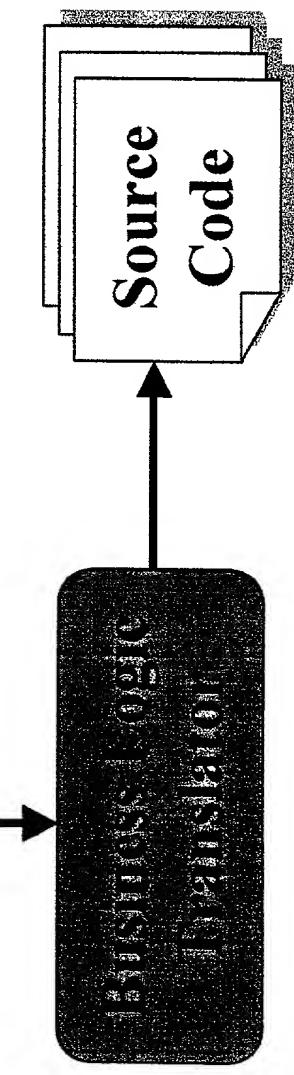
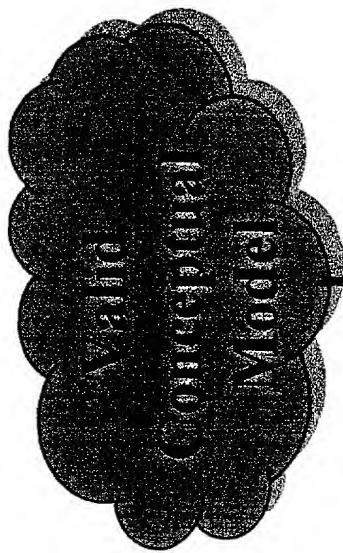
Index

- Intro
- Overview
- Output Detail
- Translation
 - CM Subset of Interest
 - Translation Processes
- Example

Intro

- Business Logic Translation is the process to obtain, following a precise Execution Model, the source code corresponding to the business logic from a valid Conceptual Model for a target Programming Language and Software Architecture.
- Execution Model is independent from Programming Language and Software Architecture.

Overview



Determines:

-Target Programming Language

-Target Software Architecture

Output Detail

- Target Programming Language and Software Architecture determine:
 - Source code organization in files
 - Files internal organization
- Source Code's backbone: Execution Model.

Output Detail

- Traceability: Source code highly readable and maintainable thanks to:
 - Source code is always organized and structured in the same way.
 - Naming conventions applied.
 - Source code includes analysis information from the Conceptual Model as comments.

Output Detail

- Implementation of a precise Execution Model grants Functional Equivalence with Conceptual Model.
- Programming Interface to Clients for:
 - Actor Validation and Authentication.
 - Services Execution.
 - Queries Execution.
- Manages:
 - Concurrency.
 - Transactions.
 - Interoperable Objects Persistence.

Translation

- Conceptual Model Subset of Interest
 - Object Model
 - Static properties (Visibility & Persistence)
 - Attributes + Identification Functions
 - Derivations
 - Aggregation Relationships
 - Inheritance Relationships
 - Services (Execution Model)
 - Arguments
 - Preconditions
 - Transaction Formulas
 - Actors (Execution Model)
 - Integrity Constraints (Execution Model)

© SOSY Inc. 2001 Patent pending

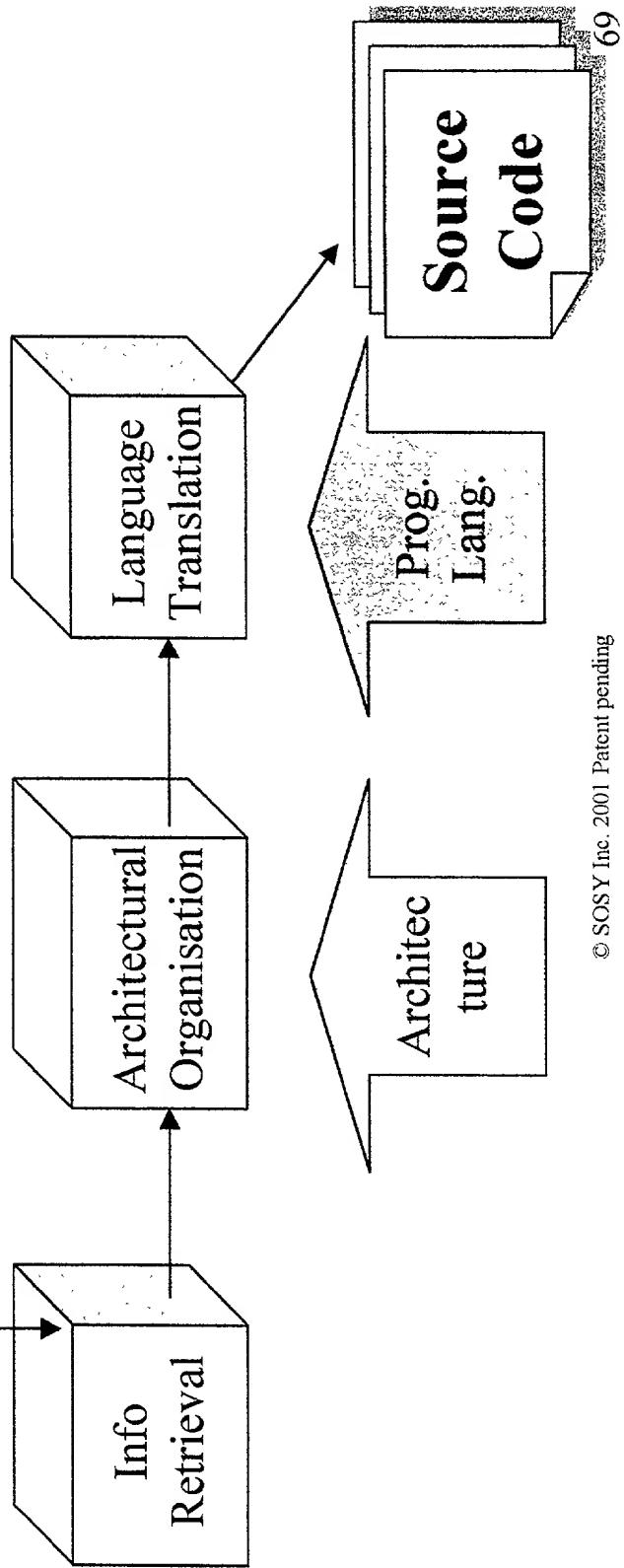
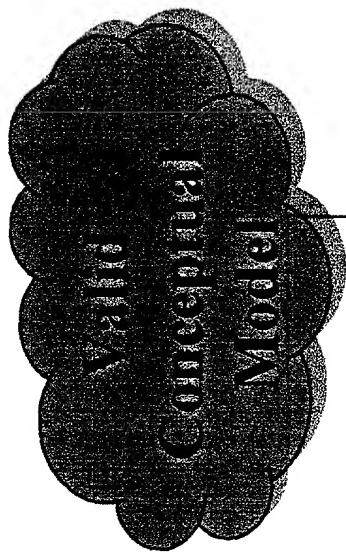
Translation

- Conceptual Model Subset of Interest.
 - Dynamic Model.
 - State Transition Diagram (Execution Model).
 - Controls Valid Lifes for an Object.
 - Object Interaction Diagram.
 - Triggers (Execution Model).
 - Global Transactions (Execution Model).
 - Functional Model (Execution Model).
 - Object state change upon occurrence of an event.

Translation

- Translation phases:
 - Information retrieval
 - Independent from target Software Architecture and Programming Language
 - Architectural organisation
 - Depends on target Software Architecture
 - Independent from target Programming Language
 - Determines files organisation and files internal structure
 - Language translation
 - Depends on target Programming Language
 - Influenced by Software Architecture
 - Takes advantage of Programming Language capabilities

Translation Phases



Translation

- Translation Processes
 - Classes
 - Static properties translation
 - Services translation
 - Queries translation
 - Global Interactions
 - Services translation
 - Global Functions
 - Functions Interface translation
 - Body is left blank

Example

- Evaluation:
 - Service Authorize modifies attributes Status, AuthoDate and AuthoComments
 - Formal Specification Language expression for evaluation Valuation
 - Authorize ()] Status=2 and AuthoDate=today() and AuthoComments="";
 - Visual Basic Produced

```
Private Function MV_Hotel_Expense_Authorize() As String
Expense_Status = 2
Expense_AuthoDate = today()
Expense_AuthoComments = ""
If MV_Hotel_Expense_Authorize = "N" Then
    End Function
```

© SOSY Inc. 2001 Patent pending

User Interface Translation

CARE Technologies, S.A.